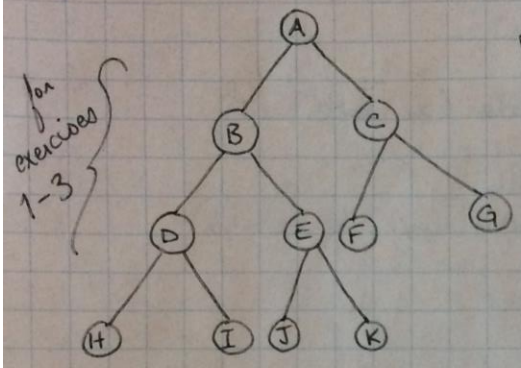
Compute the worst Case Time Complexity of the traversal methods added to LinkedSimpleTree280<I>:

Exercise 1: print the nodes in pre-order traversal:



for exercises 1-3

pre-order => root, left, right

depth first, pre-order traversal:
A, B, D, H, I, E, J, K, C, F, G

```
depthFirstPreOrder(Node cur) {        → begins at root
    print cur.item()
    if (cur.hasLeftChild())
        depthFirstPreOrder(cur.leftChild)

    if (cur.hasRightChild())
        depthFirstPreOrder(cur.rightChild)
}
```

worst case = 5 statements
best case = 3 statements
T is constant, O(T·k)
= O(k) = O(n)

Exercise 2: in-order traversal:

in-order => left, root, right

depth first, in-order traversal:
H, D, I, B, J, E, K, A, F, C, G

```
depthFirstInOrder(Node cur) {        → begins at root
    if (cur.hasLeftChild())
        depthFirstInOrder(cur.leftChild)

    print cur.item()
    if (cur.hasRightChild())
        depthFirstInOrder(cur.rightChild)
}
```

worst case = 5 statements
best case = 3 statements
T is constant, O(T·k)
= O(k) = O(n)

Exercise 3: Post-order Traversal

post-order ⇒ left, right, root

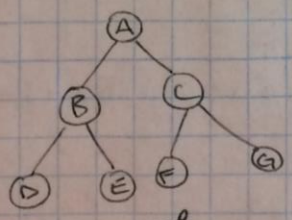
depth first, post-order traversal:
H, I, D, J, K, E, B, F, G, C, A

```
depthFirstPostOrder(Node cur) {        // begins at root
    if (cur.hasLeftChild)
        depthFirstPostOrder(cur.leftChild)
    if (cur.hasRightChild)
        depthFirstPostOrder(cur.rightChild)
    print cur.item()
}
```

worst case = 5 statements
best case = 3 statements
∴ T is constant and there are k recursive calls
O(I·k) = O(k) = O(n)

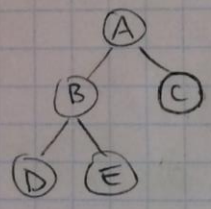
Exercise 4: Count nodes in post-order - depth first

```
int countNodesPostOrder(Node cur) {    // begins w cur = root
    int leftCount, rightCount
    if (!cur.hasLeftChild())
        leftCount = 0
    else
        leftCount = countNodesPostOrder(cur.leftChild)
    if (!cur.hasRightChild())
        rightCount = 0
    else
        rightCount = countNodesPostOrder(cur.rightChild)
    return 1 + leftCount + rightCount
}
```



T_p^R worst case & best case = 8 statements, ∴ T is constant
there will be k recursive calls, so O(I·k)
= O(k)
or O(n)

Exercise 5: Count height of tree post order → begins at root



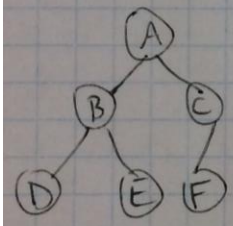

```
int treeHeightPostOrder(Node cur) {
    int leftHeight, rightHeight
    if (!cur.hasLeftChild())
        leftHeight = 0
    else
        leftHeight = treeHeightPostOrder(cur.leftChild)
    if (!cur.hasRightChild())
        rightHeight = 0
    else
        rightHeight = treeHeightPostOrder(cur.rightChild)
    return 1 + max(leftHeight, rightHeight)
}
```

worstCase: best case = Θ, $\therefore T_p^R$ is constant

$O(I_p^R \cdot k) \Rightarrow k$ is # of recursive calls

$O(k) = O(n)$

Exercise 6: Print Nodes in level order




```
printLevelOrder(T){
    if (T is empty) {
        print "T is empty"
        return
    }
    List L<node> = new List<node>()
    L.addLast(T.root)
    while (!L.isEmpty()) {
        Node toPrint = L.removeFirst()
        print toPrint.item()
        if (toPrint.hasLeftNode())
            L.addLast(toPrint.leftNode)
        if (toPrint.hasRightNode())
            L.addLast(toPrint.rightNode)
    }
}
```

worst case:

$3 + 2 + 7(n)$

$= 5 + 7(n) = O(n)$

best case:

$3 + 5(n) = O(n)$

$= \Theta(n)$